KTH Engineering Sciences

Laboration 2

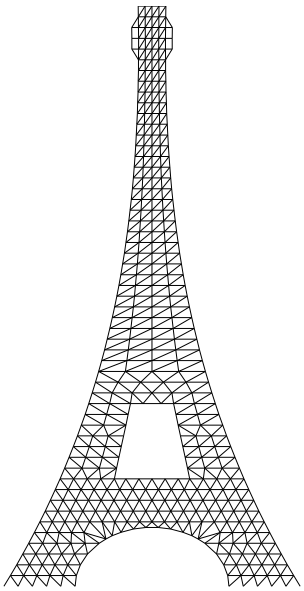
Linjära ekvationssystem och numerisk integration

Notera numeriska resultat och producera efterfrågade plottar. På redovisningen ska ni också kunna svara på frågepunkterna (•) och förklara hur era MATLAB*-program fungerar. Båda i laborationsgruppen ska kunna redogöra för teori och algoritmer! Kom väl förberedd. Ordna programfiler så att redovisningen går smidigt.*

1. Stora matriser

I många realistiska tillämpningar måste man lösa *stora* linjära ekvationsystem, med miljontals obekanta. Det är i dessa fall som effektiva algoritmer blir viktiga att använda.
Som exempel ska ni här räkna på ett komplicerat fackverk: en modell av Eiffeltornet. Ett fackverk består av stänger förenade genom leder i ett antal noder. Ni ska beräkna deformationen av fackverket när noderna belastas av yttre krafter. Ekvationerna för deformationen härleds i hållfasthetsläran, och baseras på att förskjutningarna i varje nod är små, och att Hookes lag gäller för förlängningen av varje stång.
I slutändan får man ett linjärt ekvationssystem på formen $A\boldsymbol{x} = \boldsymbol{b}$. När antalet noder i fackverket är N kommer antalet obekanta vara $2N$ och $A \in \mathbb{R}^{2N\times 2N}$. Matrisen A brukar kallas *styvhetsmatrisen.* Högerledet $\boldsymbol{b}$ innehåller de givna yttre krafterna som verkar på noderna,

$$\boldsymbol{b} = \Big(F_1^x, F_1^y, F_2^x, F_2^y, \ldots, F_N^x, F_N^y\Big)^T, \qquad \boldsymbol{b} \in \mathbb{R}^{2N},$$



Modellen i `eiffel2.mat`, med 399 noder (798 obekanta).

där $\boldsymbol{F}_j = (F_j^x, F_j^y)^T$ är kraften i nod j. Lösningen $\boldsymbol{x}$ innehåller de resulterande (obekanta) förskjutningarna,

$$\boldsymbol{x} = \Big(\Delta x_1, \Delta y_1, \Delta x_2, \Delta y_2, \ldots, \Delta x_N, \Delta y_N\Big)^T, \qquad \boldsymbol{x} \in \mathbb{R}^{2N}.$$

Här är alltså $(\Delta x_j, \Delta y_j)^T$ förskjutningen av nod j när fackverket belastas med krafterna i $\boldsymbol{b}$.

På kurshemsidan finns filerna `eiffel1.mat`, `eiffel2.mat`, `eiffel3.mat` och `eiffel4.mat`. De innehåller fyra olika modeller av Eiffeltornet med växande detaljrikedom ($N = 261$, 399, 561, 1592). Varje modell består av nodkoordinater i vektorerna `xnod, ynod`, stångindex i matrisen `bars` (används bara för plottningen) och styvhetsmatrisen `A`.

a) Ladda in en av modellerna i MATLAB med kommandot `load`. Hämta funktionsfilen `trussplot.m` från kurshemsidan och anropa den med `trussplot(xnod,ynod,bars)` för att plotta tornet. Välj en av noderna och belasta den med en kraft rakt högerut med beloppet ett. (Sätt $F_j^x = 1$ för något j, och resten av elementen i $\boldsymbol{b}$ lika med noll, dvs i MATLAB och `b=zeros(2*N,1); b(j*2-1)=1;`) Lös systemet $A\boldsymbol{x} = \boldsymbol{b}$ med backslash för att få fram förskjutningarna i alla punkter. Beräkna de nya koordinaterna för det belastade tornet, $x_j^{\text{bel}} = x_j + \Delta x_j$, etc.:

```
xbel = xnod + x(1:2:end); ybel = ynod + x(2:2:end);
```

Plotta det belastade tornet. Använd `hold on` för att plotta de två tornen ovanpå varandra i samma figur. Markera vilken nod ni valt.

b) Backslash-kommandot i MATLAB använder normalt vanlig gausseliminering. Undersök hur tidsåtgången för gausseliminering beror på systemmatrisens storlek genom att lösa ekvationssystemet $A\boldsymbol{x} = \boldsymbol{b}$ med ett godtyckligt valt högerled $\boldsymbol{b}$ för var och en av de fyra modellerna. Använd MATLAB-kommandot `cputime` (`help cputime` ger mer info). För att få bra noggrannhet i mätningen av CPU-tiden (speciellt om den är kort) bör man upprepa beräkningarna några gånger och ta medelvärdet. Plotta tidsåtgången mot antal obekanta N i en `loglog`-plot.

- Hur ska tidsåtgången bero på N enligt teorin? Stämmer det överens med din plot?

c) Ni ska räkna ut i vilka noder fackverket är mest respektive minst känslig för *vertikal* belastning. Börja med den minsta modellen, `eiffel1.mat`. Tag en nod i taget. Belasta den med kraften $F_j^y = -1$ (istället för F_j^x) och räkna ut resulterande förskjutningar $\boldsymbol{x}$. Notera storleken på den totala förskjutningen, dvs $||\boldsymbol{x}||$. Fortsätt med nästa nod, etc. Systematisera beräkningarna med en `for`-slinga i ert MATLAB-program och spara storleken på förskjutningen för varje nod. Ta sedan reda på vilken nod som ger störst respektive minst total förskjutning. Plotta tornet med `trussplot` och markera dessa mest och minst känsliga noder i figuren.

d) I c) löser ni samma stora linjära ekvationssystem med många olika högerled (N stycken). När matrisen är stor, vilket är fallet för de större modellerna, blir detta mycket tidskrävande. Optimera programmet genom att använda LU-faktorisering av A (MATLAB-kommandot `lu`).

Lös problemet i c) för var och en av modellerna, med och utan LU-faktorisering[1]. Bestäm totala tidsåtgången för varje fall. Sammanställ tiderna i en tabell.

- Varför går det snabbare att lösa problemet med LU-faktorisering?
- För vilken modell blir tidsvinsten störst? Varför?

e) När en matris är gles kan betydligt effektivare metoder än vanlig gausseliminering användas för att lösa ekvationsystemet. Kommandot `spy(A)` visar styvhetsmatrisens struktur. Förvissa er om att den är gles (bandad). Genom att tala om för MATLAB att matrisen är gles kommer bättre metoder automatiskt användas när backslash anropas. Detta kan ni enkelt göra här genom att skriva `A=sparse(A)`. Gå igenom beräkningarna i d) igen och undersök totala tidsåtgången när MATLAB använder dessa glesa lösare. Komplettera tabellen med två nya kolumner för de uppmätta tiderna.

- Vilken metod löser problemet snabbast? (Med/utan LU? Full/gles lösare?)
- Hur stor blir tidsvinsten mellan snabbaste och långsammaste metoden för minsta modellen?
- Vilket tar minst tid: minsta modellen med långsammaste metoden, eller största modellen med snabbaste metoden?

[1] Ni kan hoppa över de fall som tar orimligt lång tid.

2. Egenvärden

Förskjutningen i förra uppgiften beskriver jämviktsläget när noderna utsätts för yttre krafter. Allmänt kommer kraften på varje nod vid förskjutningen $\boldsymbol{x}$ ges av $\boldsymbol{F}_{\text{nod}} = -A\boldsymbol{x}$ om man bortser från friktion. I det dynamiska (tidsberoende) fallet följer $\boldsymbol{x} = \boldsymbol{x}(t)$ Newtons andra lag, vilken därför blir

$$\boldsymbol{F}_{\text{nod}} = m\frac{d^2\boldsymbol{x}}{dt^2} \quad \Rightarrow \quad m\frac{d^2\boldsymbol{x}}{dt^2} + A\boldsymbol{x} = 0,$$

där m är en effektiv massa för noderna (en liten förenkling av verkligheten). Vi antar i fortsättningen att $m = 1$.

En lösning till denna ODE är den oscillerande funktionen

$$\boldsymbol{x}(t) = \sin\left(t\sqrt{\lambda}\right)\boldsymbol{y},$$

där λ, $\boldsymbol{y}$ är ett egenvärde respektive en egenvektor till A-matrisen[2]. Detta gäller för alla par av egenvärden och egenvektorer. Egenmoderna till A beskriver de möjliga svängningarna (resonanserna) i fackverket. Egenvärdet λ motsvarar svängningens *frekvens* (i kvadrat) och egenvektorn $\boldsymbol{y}$ dess *amplitud* i varje nod.

> Den allmänna lösningen till differentialekvationen är en superposition av alla möjliga sådana svängningar:
>
> $$\boldsymbol{x}(t) = \sum_{k=1}^{2N}\left[\alpha_k \sin\left(t\sqrt{\lambda_k}\right) + \beta_k \cos\left(t\sqrt{\lambda_k}\right)\right]\boldsymbol{y}_k,$$
>
> där $\lambda_k, \boldsymbol{y}_k$ är egenvärdena/vektorerna till A och α_k, β_k är konstanter som bestäms av begynnelsedata.

a) Välj en av de mindre modellerna och använd MATLAB-kommandot `eig` för att beräkna de fyra lägsta svängningsmoderna, dvs de fyra minsta egenvärdena och motsvarande egenvektorer. (`eig` returnerar inte egenvärdena i storleksordning. Använd därför `sort`-kommandot på lämpligt sätt för att få fram dem i rätt ordning.) Plotta tornet när noderna är förskjutna med full amplitud och ange frekvensen för var och en av dessa moder. (Om inte förskjutningen syns bra, prova att multiplicera den med ett tal större än ett.)

Testa att animera svängningsmoderna med scriptet `trussanim.m`, som finns på kurshemsidan. Prova gärna fler moder.

b) För de stora modellerna tar `eig`-kommandot mycket lång tid. Vi vill ändå hitta de lägsta egenfrekvenserna till dessa. Implementera därför inversa potensmetoden. Använd den för att beräkna det minsta egenvärdet för varje modell med tio korrekta decimaler.

- Vad blir de lägsta egenfrekvenserna för de olika modellerna? Skiljer de sig mycket?
- Hur många iterationer behövs för de olika fallen?
- Hur skulle man kunna använda LU-faktorisering och metoder för glesa matriser för att optimera inversa potensmetoden och minska beräkningskostnaden?

[2] Derivering ger att $\boldsymbol{x}''(t) = -\lambda\boldsymbol{x}(t)$. Eftersom $\boldsymbol{y}$ är en egenvektor har vi också att $A\boldsymbol{x}(t) = \lambda\boldsymbol{x}(t)$. Således är $\boldsymbol{x}''(t) + A\boldsymbol{x}(t) = 0$ och $\boldsymbol{x}(t)$ löser ODEn.

3. Numerisk integration

Konturen för en rotationssymmetrisk lur definieras av funktionskurvan

$$y(x) = \frac{1 - \frac{1}{\pi}\arctan[p(x-1)]}{2 - \cos(\pi x)}, \quad 0 \le x \le L,$$

där p är en formparameter och L är lurens längd. Luren uppstår genom att kurvan roteras kring x-axeln och rotationsvolymen är

$$V = \pi \int_0^L y^2 \, dx.$$

Vi önskar beräkna volymen för en lur med längden $L = 2.6$ och $p = 1$.

a) Implementera de sammansatta versionerna av trapetsregeln och Simpsons formel. Beräkna integralen med båda metoderna, för de två steglängderna $h = 0.1$ och $h = 0.05$.

- Hur många decimaler verkar tillförlitliga i resultaten för de två metoderna?
- Vad hade kunnat sägas om tillförlitligheten om man bara beräknat integralen med en steglängd?

b) Undersök nu mer precist hur approximationsfelet E_h för metoderna beror på steglängden h.

1. Plotta E_h som funktion av h för trapetsregeln och Simpsons metod i samma figur. Använd MATLABs kommando `loglog`, gärna tillsammans med `grid`-kommandot. (Gör först en mycket noggrann referenslösning med Simpsons metod som ni använder för att beräkna felet på liknande sätt som i Lab 1.) Notera att antal punkter måste vara udda i Simpson-fallet.

   - Uppskatta båda metodernas noggrannhetsordning med hjälp av figuren. Stämmer det med teorin?

2. Beräkna för en följd av små h kvoten

   $$\frac{I_h - I_{h/2}}{I_{h/2} - I_{h/4}},$$

   där I_h är integralapproximationen med steglängd h.

   - Uppskatta noggrannhetsordningen för metoderna från dessa värden. (Se föreläsningsanteckningarna om noggrannhetsordning.)

c) Ni ska nu jämföra er implementation av Simpsons metod med MATLABs inbyggda integrallösare `quad` som använder en adaptiv version av Simpson. (Gör `help quad` för mer info.) Beräkningskostnaderna för metoderna bestäms typiskt i första hand av antalet funktionsevalueringar. För att jämförelsen ska bli rättvis ska ni därför göra den på följande vis:

Beräkna först integralen med `quad` och toleransen `tol`=10^{-6}. Notera resultatet samt hur många funktionsevalueringar `quad` gör. (Det ges av andra returargumentet.) Beräkna sedan integralen med er Simpson-implementation där ni använder *lika många* funktionsevalueringar, dvs punkter (som måste vara udda!). Beräkna felen i de två uträknade värdena genom att använda en referenslösning (som lämpligen också beräknas med `quad`, men med mycket hög tolerans, tex `tol`=10^{-14}). Upprepa proceduren med formparametern satt till $p = 1000$. (Notera att en ny referenslösning behövs.)

- Vilken av metoderna är effektivast i de två fallen, dvs vilken ger minst fel per funktionsevaluering?
- Hur ändras den relativa effektiviteten hos metoderna när p ändras? Varför?
  Tips: Plotta integranden för de två olika p-värdena.

En fin tredimensionell lurbild gör man så här: Låt `x` och `f` vara kolumnvektorer för konturkurvan $y(x)$. Skapa en radvektor för rotationsvinkeln $0 \leq \varphi \leq 2\pi$ med lagom steg, tex $2\pi/30$. Bilda matriser `X`, `Y` och `Z`:

```
X=x*ones(size(fi)); Y=f*cos(fi); Z=f*sin(fi);
```

Skriv `mesh(X,Y,Z)` som ger en nätfigur eller välj `surf(X,Y,Z)` eller `surfl(X,Y,Z)` som ger en fylld 3D-figur (gör gärna `help surfl`). Testa andra möjligheter: `axis equal`, `axis off`, `shading interp`, etc.

SF1544, Numeriska metoder gk IV

Laboration 2 redovisad och godkänd! Datum: .....

Namn:..... Godkänd av .....